Amendments to the Claims:

1. (cancelled)

2. (currently amended) The method of claim ~~1~~ 56, wherein the data store is a relational database management system.

3. (currently amended) The method of claim ~~1~~ 56, wherein the data store is an XML database management system.

4. (currently amended) The method of claim ~~1~~ 56, wherein the data store is a file system.

5. (currently amended) The method of claim ~~1~~ 56, wherein the at least one application state comprises a representation of a runtime snapshot of application under test which defines the context of external interaction.

6. (previously presented) The method of claim 5, wherein the at least one application state comprises a set of application objects, attributes of the application objects, and values of the attributes.

7. (previously presented) The method of claim 5, wherein the at least one application state is a plurality of application states, and the plurality of application states are arranged in a hierarchical manner.

8. (currently amended) The method of claim ~~1~~ 56, wherein the at least one external interaction sequence comprises a representation of events invoked by at least one external agent on the set of application objects.

9. (previously presented) The method of claim 8, wherein the at least one external agent is at least one of a human agent or a software agents.

10. (previously presented) The method of claim 8, wherein the at least one interaction sequence comprises at least one flow control structure for capturing at least

one of a sequential interaction, a concurrent interaction, a looping interaction, or a conditional interaction.

11. (currently amended) The method of claim ~~1~~ 56, wherein the validating step comprises both internal validation and external validation.

12. (previously presented) The method of claim 11, wherein the internal validation ensures that components of the at least one particular test case, the at least one external interaction sequence, and the input data are consistent with each other and with an application object model.

13. (previously presented) The method of claim 12, wherein the application object model comprises a metadata representation for modeling an application under test.

14. (previously presented) The method of claim 13, wherein the metadata representation comprises object type definitions for application objects.

15. (previously presented) The method of claim 13, wherein the metadata representation comprises attribute definitions for each type of application object.

16. (previously presented) The method of claim 13, wherein the metadata representation comprises a definition of methods and events that are supported by each type of application object.

17. (previously presented) The method of claim 13, wherein the metadata representation comprises a definition of effects of events on the at least one application state.

18. (previously presented) The method of claim 14, wherein the object type definitions further comprise categorization of each type of application object as a hierarchical type, a container types, or a simple type.

19. (previously presented) The method of claim 18, wherein a particular application state is associated with each hierarchical object type, and wherein a container type is an application object types that can contain other application objects.

20. (previously presented) The method of claim 19, wherein the particular application state associated with a hierarchical object type is at least one of a modal application state or a nonmodal application state.

21. (previously presented) The method of claim 20, wherein the modal application state restricts possible interactions to application object instances available within the at least one application state.

22. (previously presented) The method of claim 17, wherein the effects of events on an application state capture one or more consequences of the event to the application state.

23. (previously presented) The method of claim 22, wherein a consequence of an event is selected from, the set consisting of: creation of a new object instance of a given type, deletion of an object instance of a given type, modification of attributes of an existing object instance, and selection of an instance of an object type.

24. (previously presented) The method of claim 23, wherein creation of a new object instance of a hierarchical object type results in creation of a new application state.

25. (previously presented) The method of claim 23, wherein the selection of an object instance of a hierarchical object type results in selection of a certain application state associated with that object instance.

26. (previously presented) The method of claim 11, wherein the external validation validates the at least one rule-based generated test case against an application metadata repository.

27. (previously presented) The method of claim 26, wherein the application metadata repository comprises definitions of application objects and specification of interactions of the application objects within the application under test.

28. (previously presented) The method of claim 26, wherein the external validation provides a static verification test for the at least one rule-based generated test cases.

29. (previously presented) The method of claim 26, wherein the external validation increases productivity by identifying invalid test cases.

30. (previously presented) The method of claim 26, wherein the external validation increases productivity by identifying inconsistencies in statically verifiable application behaviors.

31. (currently amended) The method of claim ~~1~~ 56, wherein the at least one test scripts comprises a rule-based test cases represented in a scripting language.

32. (previously presented) The method of claim 31, wherein the scripting language is at least one of a typed programming language or an untyped programming languages used for at least one of recording or authoring test cases.

33. (currently amended) The method of claim ~~1~~ 56, further comprising:

providing rules for selection of components of test case definitions, external interaction sequences and input data; and further providing rules for data driven test case generation.

34. (previously presented) The method of claim 33, wherein the selection rules are specified using at least one query languages.

35. (previously presented) The method of claim 34, wherein the at least one query language is Structured Query Language (SQL).

36. (previously presented) The method of claim 34, wherein the at least one query language is Extensible Markup Language Query (XQuery) language.

37. (previously presented) The method of claim 34, wherein the at least one query language is Application Programming Interface (API) called from code written in a programming language.

38. (previously presented) The method of claim 34, wherein the use of the at least one query languages allows test cases to be generated from live customer data.

39. (previously presented) The method of claim 33, wherein the data driven test case generation comprises composing the test case based on the input data.

40. (previously presented) The method of claim 39, wherein the input data comprises a plurality of datasets, and wherein at least one of a plurality of test cases or a plurality of external interaction sequences repeated within a loop control structure is generated for each dataset.

41. (previously presented) The method of claim 39, wherein a portion of the input data comprises a plurality of datasets, and the interaction sequences corresponding to the portion of input data are repeated within a loop control structure.

42. (previously presented) The method of claim 39, wherein each element of input data is flagged as either valid or invalid.

43. (previously presented) The method of claim 42, wherein the generation step further comprises providing an appropriate interaction sequence for exception handling when the presence of a first validity flag in an element of the input data that is different from a second validity flag corresponding to the input data when the at least one particular test cases was at least one of recorded or authored.

44. (currently amended) The method of claim ~~1~~ 57, further comprising:

using a language mapping to generate the at least one test script in at least one scripting language.

45. (previously presented) The method of claim 44, wherein the language mapping maps external interactions captured as events on an application object to particular statements in the scripting language.

46. (previously presented) The method of claim 44, wherein a plurality of language mappings are provided at the same time.

47. (previously presented) The method of claim 44, wherein a plurality of test scripts is converted using a plurality of scripting languages at the same time.

48. (previously presented) The method of claim 47, wherein the plurality of test scripts can be used in a plurality of test execution environments.

49. (cancelled).

50. (currently amended) The system of claim ~~49~~ 57, wherein the third set of instructions, when executed by the processor, further configures the processor to verify that components of the at least one test script, the at least one external interaction sequences~~,~~ and the input data are each consistent with each other and with an application object model.

51. (currently amended) The system of claim ~~49~~ 57, wherein the third set of instructions uses is external validation logic.

52. (previously presented) The system of claim 51, wherein the third set of instructions, when executed by the processor, further configures the processor to validate the at least one test script against an application metadata repository.

53. (currently amended) The system of claim ~~49~~ 57, further comprising:

a fifth set of instructions which, when executed by the processor, configures the processor to provide rules for selection of components of test case definition, external interaction sequences and input data; wherein and the rules are data driven test case generation.

54. (currently amended) The system of claim ~~49~~ 57, further comprising:

a sixth set of instructions which, when executed by the processor, configures the processor to provide data driven test case generation.

55. (previously presented) The system of claim 54, wherein the sixth set of instructions, when executed by the processor, further configures the processor to compose the at least one rule-based test case as dictated by the input data.

56. (new) A method for generating test scripts comprising:

providing at least one particular test case in a data store in an abstract representation form;

using abstract representations that have at least three separate components including at least one application state, at least one external interaction sequence, and at least one input data, the at least one application state having at least one of: (a) a set of application objects associated with a set of attributes and their values, or (b) a runtime snapshot of an application under test which defines a context of external interaction;

selecting at least one test case in its abstract representation and using rules for the selection of the at least one application state, the at least one external interaction sequence and the input data, and using the rules to validate the at least one rule-based test case against an application object model, where the application object model is a metadata representation for modeling application under test and includes components selected from application object type definitions for application objects, attribute definitions for each application object type, definitions of methods and events that are supported by each application object type and definitions of effects of events on an application state;, and

generating at least one test script based on the at least one rule-based selected test wherein the at least one test script can be run in a particular one of a plurality of target test execution environments.

57. (new) A computer system for generating test scripts, comprising:, comprising:

a processor;

a memory arrangement coupled to the processor, the memory arrangement configured to store at least one particular test case in a data store in an abstract representation form;

a first set of instructions which, when executed by the processor, configures the processor to use abstract representations that have at least three separate components including at least one application state, at least one external interaction sequence, and at least one input data, the at least one application state having at least one of: (a) a set of application objects associated with a set of attributes and their values, or (b) a runtime snapshot of an application under test which defines a context of external interaction;

a second set of instructions which, when executed by the processor, configures the processor select at least one test case in its abstract representation and using rules for the selection of the at least one application state, the at least one external interaction sequence and the input data, and using the rules to validate the at least one rule-based test case against an application object model, where the application object model is a metadata representation for modeling application under test and includes components selected from application object type definitions for application objects, attribute definitions for each application object type, definitions of methods and events that are supported by each application object type and definitions of effects of events on an application state

a third set of instructions which, when executed by the processor, configures the processor to generate at least one test script based on the at least one rule-based selected test wherein the at least one test script can be run in a particular one of a plurality of target test execution environments.